(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 1 021 021 A2

AB

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
19.07.2000 Bulletin 2000/29

(51) Int. Cl.[7]: **H04L 29/06**

(21) Application number: **00300148.4**

(22) Date of filing: **11.01.2000**

(84) Designated Contracting States:
AT BE CH CY DE DK ES FI FR GB GR IE IT LI LU MC NL PT SE
Designated Extension States:
AL LT LV MK RO SI

(30) Priority: **13.01.1999 US 231081**

(71) Applicant:
International Business Machines Corporation
Armonk, NY 10504 (US)

(72) Inventors:
- **Malkin, Peter Kenneth**
  Winchester, Hampshire SO21 2JN (GB)
- **Yu, Philip Shi-Lung**
  Winchester, Hampshire SO21 2JN (GB)

(74) Representative:
Jennings, Michael John
IBM United Kingdom Limited,
Intellectual Property Department,
Hursley Park
Winchester, Hampshire SO21 2JN (GB)

(54) Method and apparatus for providing awareness-triggered push

(57) The present invention enables, in a network, state notifications to trigger data transfer from a source server or from an intermediate proxy server to an often disconnected client computer, based on the client's entering a particular state. The intermediate proxy server queues push requests from source servers to target clients. The state notifications include state information based on different criteria. In the state notifications may specify push information which may be used to determine the least expensive device to which to transfer given data. The state notifications also reduce the network polling load by initiating each poll.

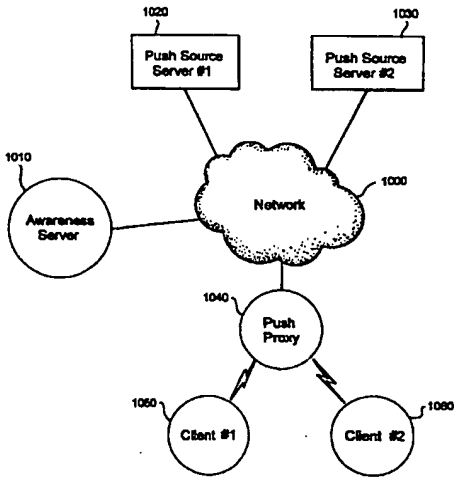


Fig. 1

EP 1 021 021 A2

Description

BACKGROUND OF THE INVENTION

Field of the Invention

[0001] The present invention is related to data transmission in a computer network, and more particularly to a unique data push technology where data is transmitted to recipients according to a particular state of the intended recipients.

Description of Prior Art

[0002] The standard client-server information push model is not efficient for pervasive computing (PvC) entities, which for the most part are disconnected from the network. Servers uselessly attempting to push data to disconnected clients waste resources, such as the CPU, the RAM, and the network bandwidth in the process. Example PvC entities are palm-sized computing devices (Personal Digital Assistants such as PalmPilot from 3Com, and smart phones), laptop computers, embedded computing devices within other apparatus (such as in a car or television set, or controlling a remote sensor in a gas pipeline), and smaller devices such as may be embedded in clothing.

Push

[0003] In a standard push model, such as HTTP Push, a given push source server (PSS) sends out new information to all subscribed clients. This works well in networks where clients are actively connected to the network most of the time and the network has sufficient bandwidth for the transmission. Such clients are able to receive the transmissions. However, because PvC devices are mostly disconnected from the network, this model fails since the majority of the transmissions will fail.

[0004] Push source servers may attempt to avoid transmissions to disconnected clients by checking the availability of the target clients before each transmission. This approach still results in heavy network load due to the large number of status checking probes.

Pull

[0005] If any given client machine was forced to request all information sent to it, e.g., using the HTTP GET request, then every data transmission would be guaranteed to be sent to an available client. The problem with the pull scenario is that because there is no way for a given client to know when or if a given source server has any new information, the client is forced to make many unnecessary pull requests to ensure that any and all new information is retrieved. This will result in a heavy network load due to the large number of unnecessary pull requests.

Poll

[0006] Yet another approach, similar to the pull technique, is polling, like that used by the PointCast Network http://www.pointcast.com, an Internet news service broadcasting personalized news and information directly to clients' computer screens. In polling, just as with the push approach, additional network load is created by clients having to continually request information from source servers, even when no new information is in fact available, to ensure that any and all new data is retrieved.

[0007] Since PvC entities are already bandwidth-challenged, there remains a need for a way to enable an automated delivery of information, especially for that which is critical, yet only infrequently provided, such as bug fixes or virus checker updates.

SUMMARY OF THE INVENTION

[0008] The present invention provides a method and an apparatus for enabling a network server to trigger the transfer of data from a source server to a client entity, especially such that is often disconnected, like a typical PvC device, based on the client entity's particular state. The client entity's state may be the state of being connected to the network, for instance. The network server, which will be referred to as an "awareness server," preferably provides access to the transient status information of the network client entities, such as users, devices, applications, etc. The status information preferably specifies whether a given device is connected to the network, a given user is logged on, and whether a given user is actively participating in a given application. The process by which the awareness server triggers data transfers preferably comprises the following steps:

the source server requesting the awareness server to notify it when a specific client entity is in a particular state;

the client entity being in or entering the particular state;

the awareness server sending a notification to the requesting server after learning of the specific client entity's state; and

the source server receiving the notification and transferring data to the specific client entity.

[0009] The awareness server may also initiate data transfers from an intermediate proxy server in which push requests from source servers to target client entities are queued. Other methods are also provided to

enable awareness notifications to be generated based on different criteria including:

whether a given client entity is connected and the manner of the connection, for example, the target client entity becomes TCP/IP linked at speeds of 14.4 Kbps or faster;

whether a given client entity is executing a particular application, for example, a web browser, and the manner of the execution;

whether a connection to a given client entity is secured and the manner of the connection, for example, when the target client entity becomes linked using SSL;

whether or not a given user is currently connected to the network, returning the network address of the user's current device in the notification; and

who is using a given client entity, for example, when the target client entity is being used by a particular user.

[0010] Methods are further provided for the awareness server to obtain state information by:

actively monitoring requested state and state changes;

learning of states or state changes from the target client entities themselves; and

learning of states or state changes from other servers, for example, learning from an Internet Access Provider when a given client is connected to the network.

[0011] Additional methods are provided for:

specifying information required for pushes in the awareness notifications, e.g., the port number of the application to which to transmit a given push;

determining the least expensive client device to which to transfer the given data, by using the awareness server; and

reducing the load of polling on the network, such as that done by network SNMP-based management servers, by initiating each poll by an awareness notification.

BRIEF DESCRIPTION OF DRAWINGS

[0012] The present invention may be more readily understood by one skilled in the art with reference to the following detailed description of preferred embodiments thereof, taken in conjunction with the accompanying drawings wherein like elements are designated by identical reference numerals throughout the several views, and in which:

Figure 1 is a diagram of a network having features of the present invention including a standalone push proxy;

Figure 2 is a component diagram of an awareness server;

Figure 3 is a flow diagram of the awareness server logic including the awareness request handler;

Figure 4 is a flow diagram of the awareness notification handler;

Figure 5 is a component diagram of a push proxy;

Figure 6 is a flow diagram of the push proxy logic including the proxied push request handler and the proxied push handler; and

Figure 7 is a diagram of a network having features of the present invention including server-side push proxies.

DETAILED DESCRIPTION OF THE INVENTION

[0013] In the preferred embodiment, the present invention is directed to a method and apparatus for providing HTTP PUSH transmissions. Those skilled in the art will understand that other types of transmissions, including but not limited to e-mail and multimedia broadcast may be handled similarly.

[0014] Figure 1 shows a diagram of an overall logical network topology. PSS 1020, 1030, an awareness server 1010, and a push proxy server 1040 are directly connected to the network 1000. Some examples of the network 1000 include, but are not limited to, the Internet, the World Wide Web, an Intranet and local area networks (LANs).

[0015] The client devices 1050, 1060 are locally connected to the push proxy server 1040. All communication between the clients 1050, 1060 and the network 1000, passes through the push proxy server 1040. Although only two PSS 1020, 1030 are depicted, it is understood that the present invention is applicable to any number of such servers. Similarly, the present invention is applicable to any number of clients.

[0016] The PSS 1020, 1030 are essentially standard HTTP servers including, but not limited to IBM's Internet Connection Server (ICS). The only unique feature of the PSS 1020, 1030 is that they route all pushes through a particular push proxy server 1040. This feature may be added to an ICS-based server using ICS'

API (ICSAPI), which enables the development and inclusion of modules that customize the behavior of the ICS server. An ICS server may be configured to hand off control to such a module at a particular point during the processing of requests. The module may perform its task and then either return control to the server to complete the transmission, or complete the transmission itself and notify the server when it is done.

[0017] In the case of the PSS 1020, 1030 the added module takes control before the data is finally pushed to the subscribed client 1050, 1060 and, rather than sending the data to the client 1050, 1060, connects to the push proxy 1040 and specifies where the data should be sent. This proxy routing is part of the HTTP standard specified in HTTPD RFC. The awareness server 1010, the PSS 1020, 1030 and the push proxy server 1040 may be implemented on any computing nodes including, but not limited to products, such as IBM S/390 SYSPLEX, SP2, or RS6000 workstations.

[0018] The clients 1050, 1060 may include an IBM WorkPad, a PC, various workstations, set top box, etc. The software running on the clients 1050, 1060 may include, but is not limited to, the Netscape's Navigator web browser.

[0019] The awareness server 1010 provides notification of when the specified clients or targets have reached a desired state, for example, connected to the network. As shown in Figure 2 the awareness server 1010 (Figure 1) includes a CPU 3000, a memory 3020, such as RAM, and a storage device 3010, such as a disk or other direct access storage device (DASD).

[0020] The awareness server 1010 (Figure 1) logic preferably embodied as executable computer code, is loaded in to the memory 3020, either remotely over the network or locally from an optical CD-ROM, magnetic storage, such as disk, or DASD 3010, for execution by CPU 3000. The executable computer code loaded into memory 3020 preferably includes an awareness request handler 3030 to accept and to service a clients' requests, an awareness notification handler 3040 to notify clients of the status of their requests, and a request buffer 3050 to store current requests.

[0021] As shown in Figure 3 the awareness server 1010 (Figure 1) waits for input in step 4000, possible inputs including requests for status notification received from the network. After the input is received, step 4010 determines whether the input is an awareness request, i.e., a request to notify that a specified client has reached a particular state. If the request is not a request for status notification, then a miscellaneous handler is invoked in step 4030. Following the execution of the miscellaneous handler in step 4030, the program control returns to step 4000 to wait for further input.

[0022] However, if the input is an awareness notification request, then the awareness request handler 4020 is invoked. At step 5010 the request handler reads the requestor's and the target client's IDs from the request, e.g., the hostname of the server asking to be notified and the hostname of the client device whose status is being announced. At step 5020, the awareness request handler 4020 adds a new entry to the request buffer 3050 (Figure 2) indicating the extracted IDs of the requestor and of the target client. Following the execution of the awareness request handler 4020, the program control returns to step 4000 to wait for further input.

[0023] Figure 4 shows the processing of the awareness notification handler 3040 (Figure 2) which is continually running as a background process in the CPU 3000 (Figure 2) of the awareness server 1010 (Figure 2).

[0024] A local pointer is moved to the first entry in the request buffer 3050 (Figure 2) in step 6000. The requestor's and the target client's IDs are read from the pointed to entry in step 6010 . In step 6020 the status of the target client is checked.

[0025] The status may be the connectivity status and may be checked by such programs as PING, or from a notification received from the target client device, i.e., whenever a given client device connects. Ascertaining the status of a given target client actively connected or participating in a particular application may be accomplished by requesting the target client's status from the application server. The awareness server 1010 (Figure 1) itself may request to be notified by outside servers. For example, it could learn when a given target client connects by asking the target client's Internet access provider for a notification when the given client connects. If a notification is received, then the target client is connected, otherwise the client is still disconnected.

[0026] In step 6030 a determination is made whether the target client has reached a desired status. If the target client has reached the desired status, then in step 6040 a notification is sent to the requestor using TCP, UDP or any other network transport protocol or higher level applications like e-mail. The current entry is then deleted from the request buffer 3050 (Figure 2) in step 6050. If however, the target client has not reached the needed status, then steps 6040 and 6050 are skipped.

[0027] In step 6060 if it is determined that there are no more requests in the request buffer 3050 (Figure 2), then processing continues at step 6000. If there are further requests in the request buffer 3050 (Figure 2), then in step 6070, the local pointer is advances to the next entry, and the processing continues in step 6010.

[0028] The awareness server 1010 (Figure 1) may send notifications for any number of reasons, some of these reasons may include:

whether a given client entity is connected or the manner of the connection, for example, the target client entity becomes TCP/IP linked at 14.4 Kbps or faster;

whether a given client entity is running a particular application, e.g., when the target client entity is running a web browser, or the manner of the execution of the application.

whether a connection to a given client entity is secured and the manner of the connection, for example, a client entity becomes linked using SSL.

who is using a given client entity, for example, a particular user is the client.

[0029] The awareness server 1010 (Figure 1) may also return critical information in the notifications, such as the port number on which the target client entity's push recipient application, such as a web browser, is running. This critical information may be read and used by the push proxy server 1040 (Figure 1) when the notification is received.

[0030] The push proxy server 1040 (Figure 1) shown in Figure 5, queues up push requests and executes the final transmission to the target client device after being notified by the awareness server that the given client device is able to receive the transmission. The push proxy server 1040 (Figure 1) includes a CPU 10000, a memory 10020, such as RAM, and a storage device 10010, such as a disk or a DASD.

[0031] According to the present invention, the push proxy logic embodied as executable computer code, may be loaded remotely over the network or locally from an optical CD-ROM or magnetic storage, such as disk, or DASD 10010 into memory 10020 for execution by the CPU 10000. The computer code loaded into the memory 10020 includes a proxied push request handler 10030; a proxied push handler 10040; and a proxied push database 10050 including but not limited to IBM's DB2 Database Product.

[0032] Figure 6 illustrates the push proxy program flow. As shown in Figure 6, the push proxy waits from input in step 11000. When the input is received, a determination is made in step 11010 whether the input is a PUSH request. If it is a PUSH request, then, the proxied push request handler 11020 is invoked. Otherwise, processing continues in step 11030 where it is determined whether the input is an awareness notification. If the input is an awareness notification, then the proxied push handler 11040 is invoked. Otherwise, if the input is not an awareness notification the appropriate miscellaneous handler is invoked after which the processing continues in step 11000.

[0033] The proxied push request handler 11020 queues up the push feeds from the PSS 1020, 1030 (Figure 1). The data and the target client's ID, such as the client's hostname or the IP address, are read from the request in step 12000. If it is determined in step 12010, that a given ID does not already have an entry in the proxied push database 10050 (Figure 5), then an entry is created in step 12020.

[0034] In step 12030, new data is added to the queued data for the read ID. A notification request is sent to the awareness server 1010 (Figure 1), in step 12040, specifying the ID for itself and the target client, after which the processing continues in step 11000.

[0035] As mentioned, after receiving a notification from the awareness server 1010 (Figure 1) for a specific client device, the proxied push handler 11040 pushes the queued data to that specific client device.

[0036] Particularly, at step 13000, the proxied push handler reads the client's ID from the notification, and retrieves of the data queued for the given ID from the proxied push database 10050 (Figure 5) at step 13010. Then, at step 13020, this data is pushed to the target client. In step 13030, the proxied push database entry for the given ID is deleted and the processing continues in step 11000.

[0037] Sending queued pushes only after being notified that the target client is connected to the network and is listening, eliminates many needless transmission attempts which would have been executed in the absence of this knowledge. This aspect of the present invention may also be used to optimize polling applications, for example, the status polling performed by SNMP-based network management applications. If status requests are only sent when the target network devices are connected, as indicated by notifications from an awareness server, then resource-wasting requests to disconnected devices are eliminated.

[0038] As shown in Figure 7, the push proxy server 1040 (Figure 1) need not be a standalone machine. Instead of having the push proxy run remotely, each PSS 2020, 2030 may have its own server-side proxy running concurrently in parallel with the PSS system. In this configuration, a given PSS would use its own hostname or IP address as that of the push proxy.

[0039] Multiple push proxies may be used to spread out processing and network load. With multiple push proxies, each PSS would have a choice of which one to use. Another way to spread out processing and network load is to employ multiple awareness servers, so that a push proxy may request notification from more than one awareness server.

[0040] The target client's ID sent to the awareness server 2010 could be a user ID, such as a name like "Luanda Chen", rather than a device ID, such as a network address like 192.62.1.1. Given a user ID, the awareness server could notify a push proxy when a particular user is connected, specifying the address of the user's current device in the notification.

[0041] The awareness server notifications may also be used to route transmissions as inexpensively as possible, for instance, in response to an awareness notification request for a particular user, the awareness server 2010 may determine all the currently available user devices, then determine the least expensive device to which to send data, and finally reply to the notification request with a notification which specifies the address

of this device.

Claims

1. A method to transfer data from at least one source entity, in a network, to at least one client entity in said network when said client entity has a particular operating state, said method comprising the steps of:

   a. repeatedly determining a state of said client entity until said client entity is in said particular operating state; and

   b. communicating said state to said source entity by sending a notification, wherein said source entity transfers data to said client entity in response to said notification.

2. The method of claim 1, wherein said source entity is a source server, said source server requesting said notification.

3. The method of claim 2, wherein said source entity includes an intermediary proxy, said intermediary proxy receiving a push request from a source server and requesting said notification.

4. The method of any one of claims 1 to 3, wherein data is transferred using information specified in said notification.

5. The method of any one of claims 1 to 4, wherein said particular operating state includes said client entity being connected to said network.

6. The method of claim 4, wherein said particular operating state includes a specific user being connected to said network, and wherein an address of said specific user's current device is returned in said notification.

7. The method of any one of the preceding claims, wherein said particular operating state includes said client entity running a particular application.

8. The method of any one of the preceding claims, wherein said particular operating state is determined by actively monitoring said client entity.

9. The method of claim 8, wherein said active monitoring of said client entity is performed by polling of said client entity on said network.

10. The method of any one of claims 1 to 7, wherein said particular operating state is determined by a communication from said client entity.

11. The method of any one of claims 1 to 7, wherein said particular operating state is determined by a communication from one or more entities in said network.

12. The method of any one of the preceding claims, further including a step of determining a least expensive method to transfer said data to said client entity based upon said particular operating state.

13. An apparatus to transfer data from at least one source entity, in a network, to at least one client entity in said network when said client entity has a particular operating state, said apparatus comprising:

   a. means for repeatedly determining a state of said client entity until said client entity is in said particular operating state; and

   b. means for communicating said state to said source entity by sending a notification, wherein said source entity transfers data to said client entity in response to said notification.

14. Apparatus according to claim 13 wherein said source entity is a source server, said source server requesting said notification.

15. Apparatus according to claim 14, wherein said source entity includes an intermediary proxy, said intermediary proxy receiving a push request from a source server and requesting said notification.

16. A computer program embodying a set of instructions executable by a machine to perform method steps to transfer data from at least one source entity, in a network, to at least one client entity in said network when said client entity has a particular operating state, said method comprising the steps of:

   a. repeatedly determining a state of said client entity until said client entity is in said particular operating state; and

   b. communicating said state to said source entity by sending a notification, wherein said source entity transfers data to said client entity in response to said notification.

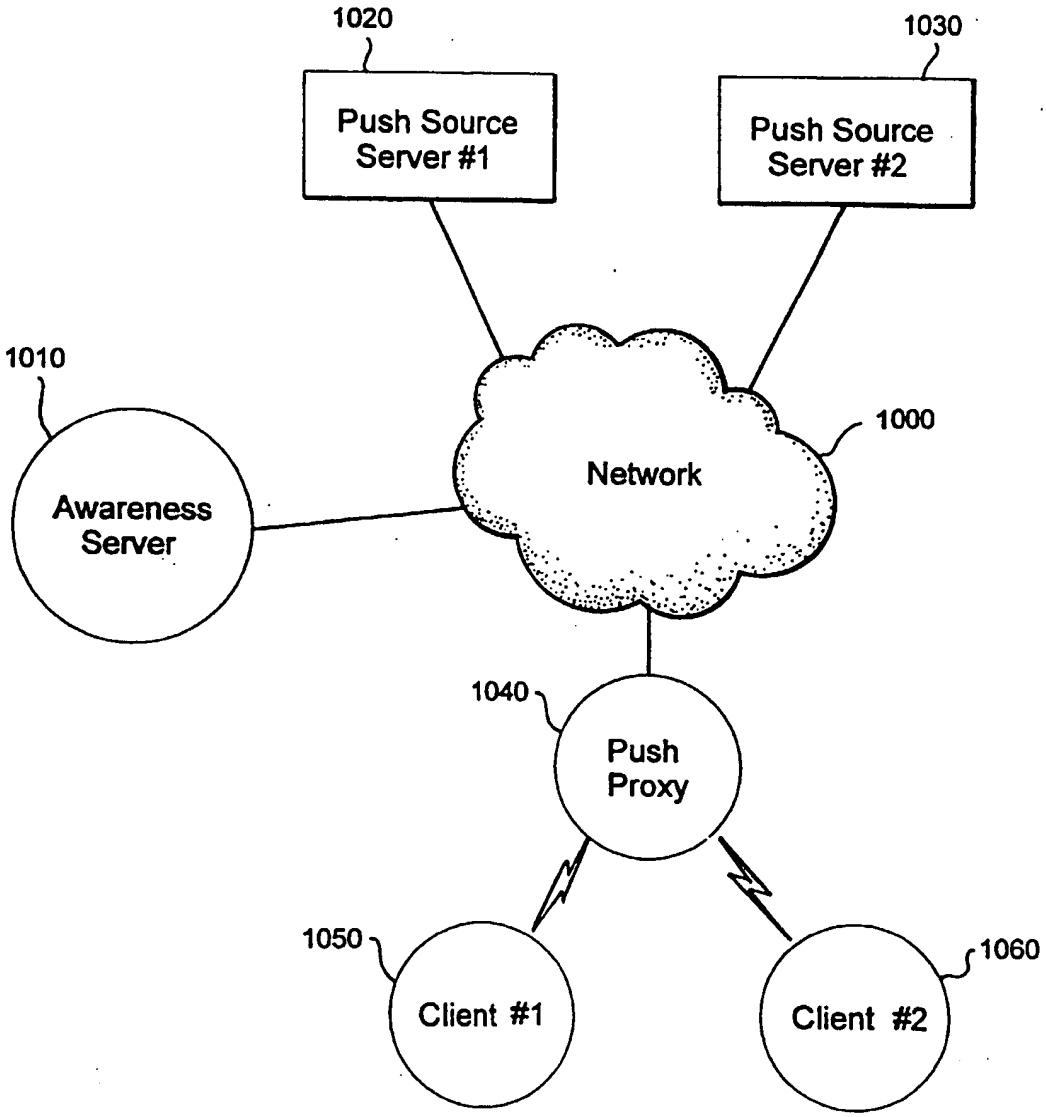

1020
Push Source Server #1
1030
Push Source Server #2
1010
Awareness Server
Network
1000
1040
Push Proxy
1050
Client #1
1060
Client #2

Fig. 1

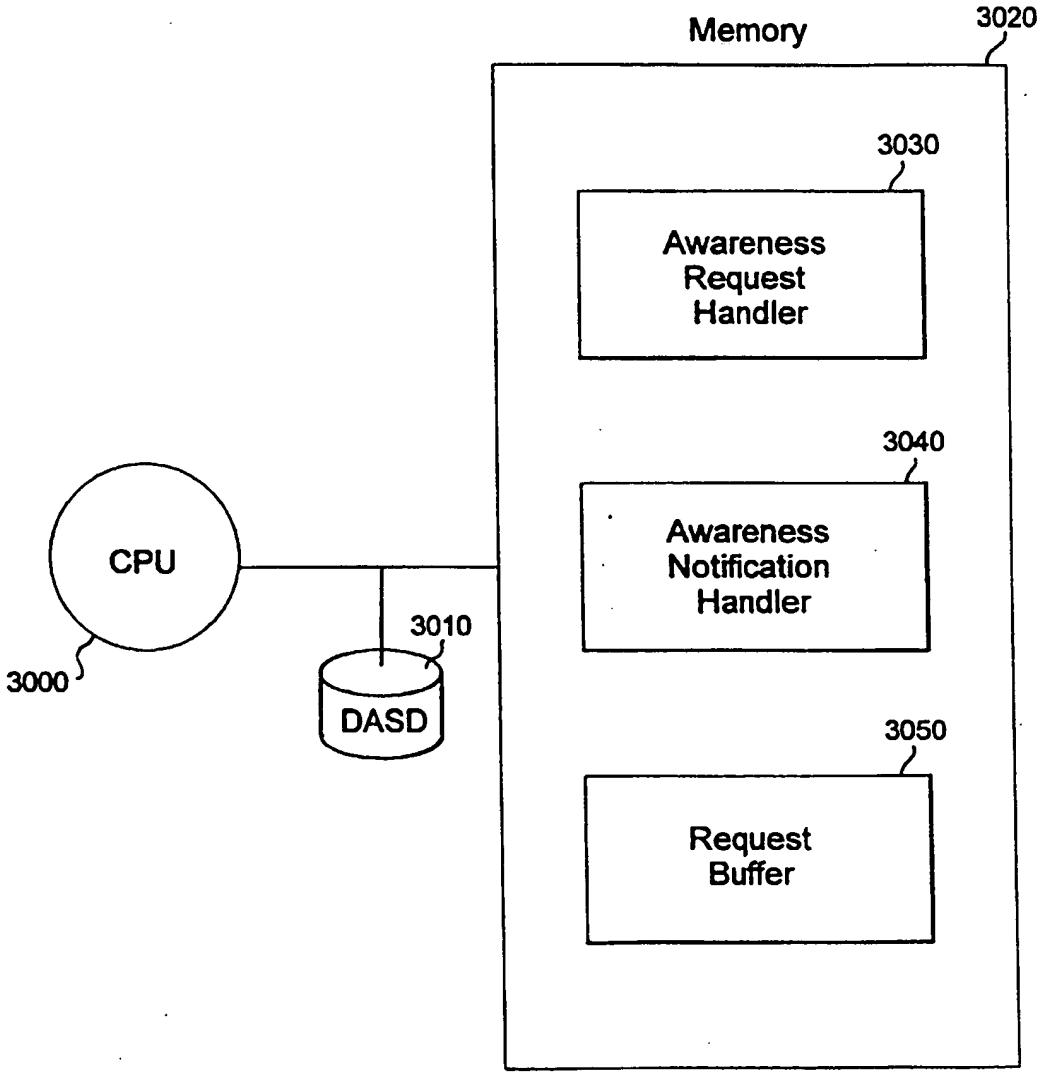

Memory
3020
3030
Awareness Request Handler
3040
Awareness Notification Handler
3050
Request Buffer
CPU
3000
3010
DASD

Fig. 2

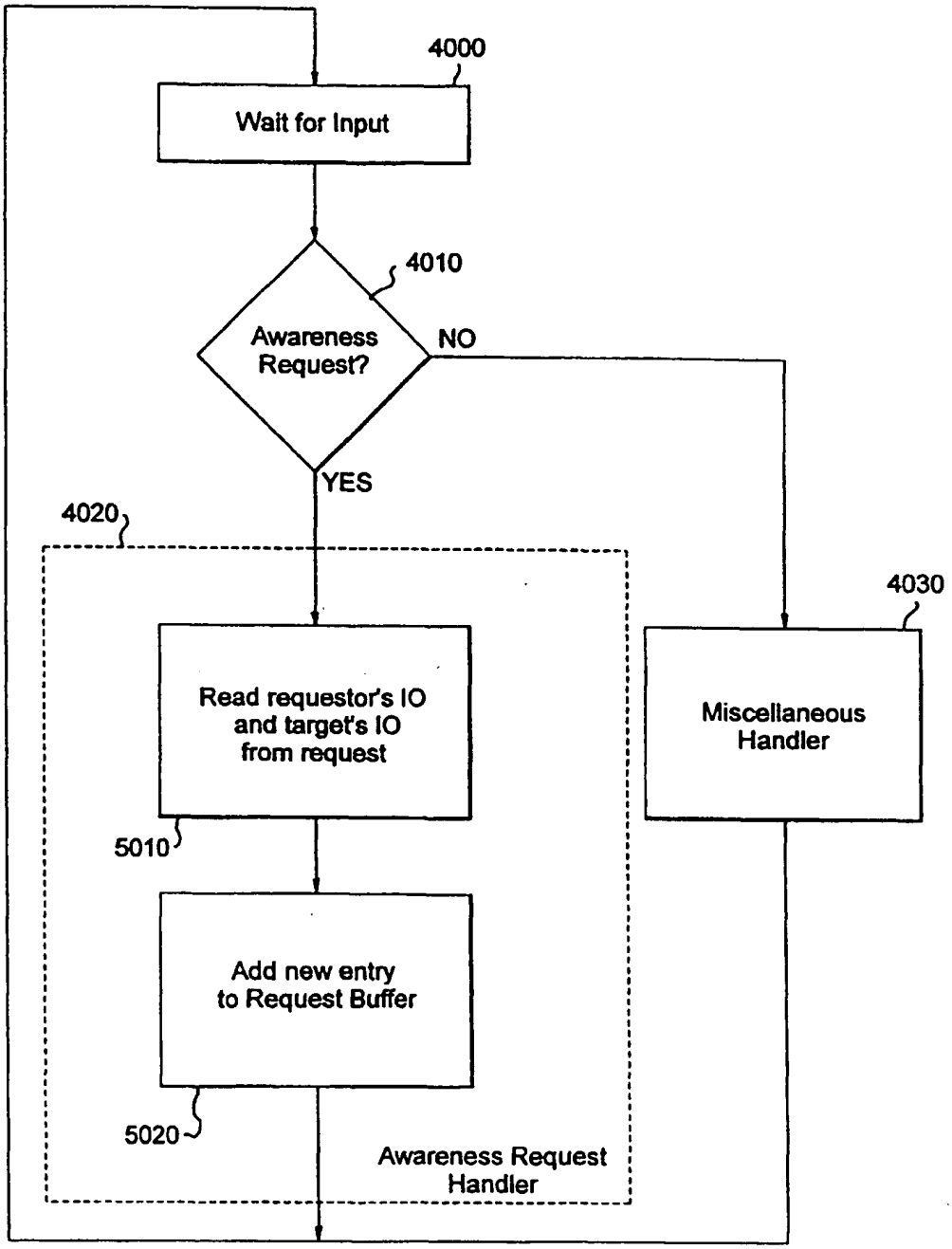

4000
Wait for Input
4010
Awareness Request?
NO
YES
4020
4030
Read requestor's IO and target's IO from request
Miscellaneous Handler
5010
Add new entry to Request Buffer
5020
Awareness Request Handler
*Awareness Notification Handler running in parallel (background)

Fig. 3

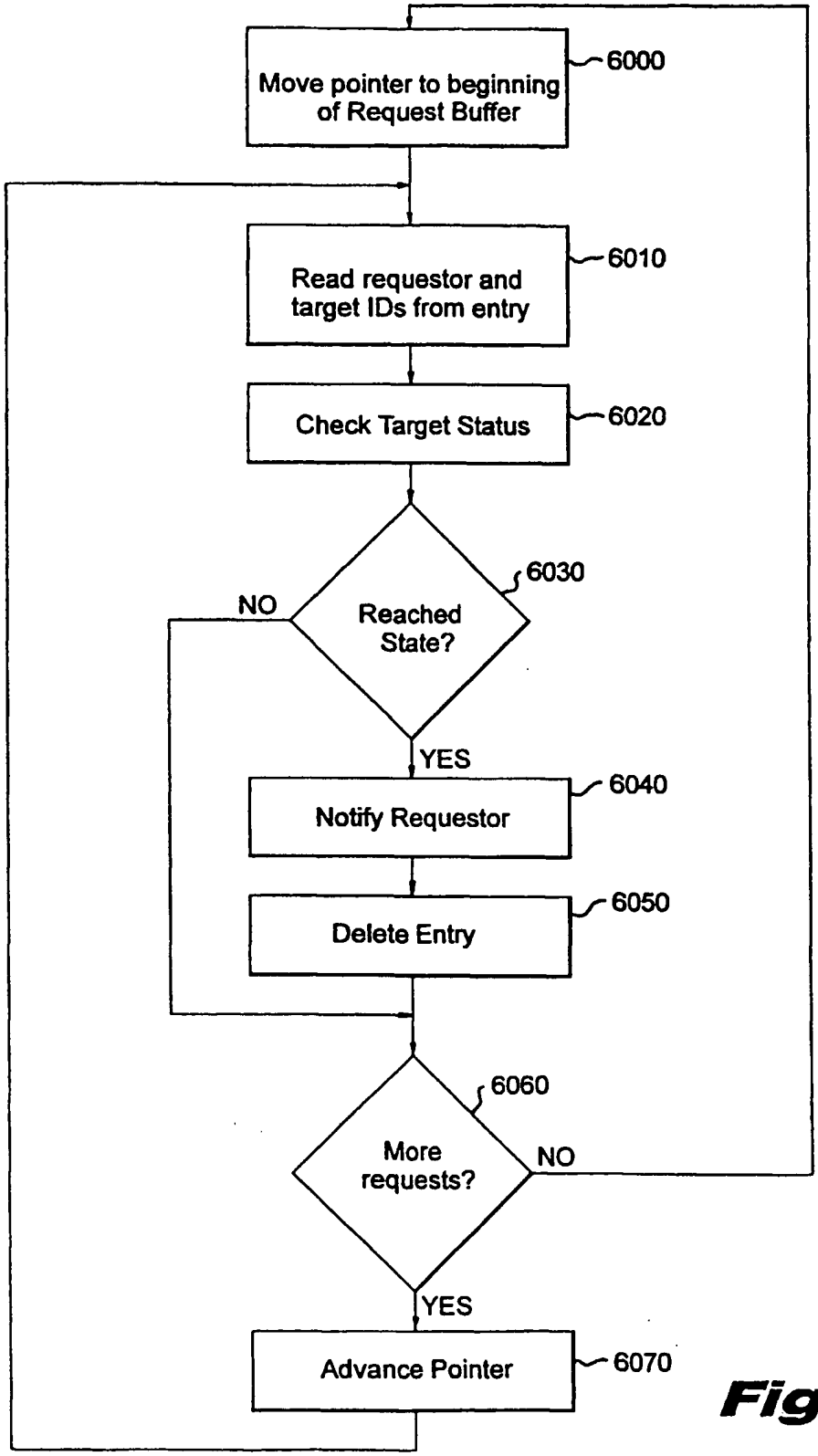

Move pointer to beginning of Request Buffer
6000
Read requestor and target IDs from entry
6010
Check Target Status
6020
Reached State?
6030
NO
YES
Notify Requestor
6040
Delete Entry
6050
More requests?
6060
NO
YES
Advance Pointer
6070

Fig. 4

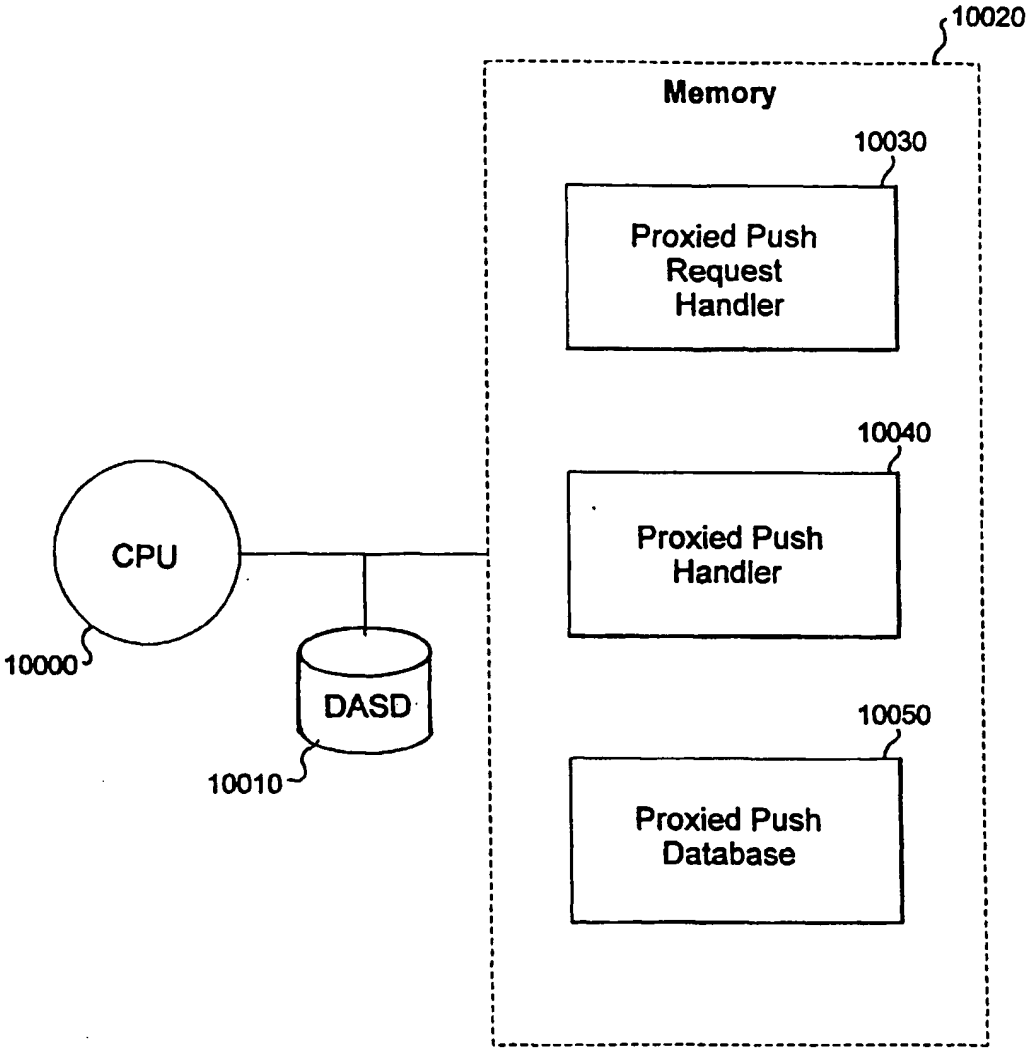
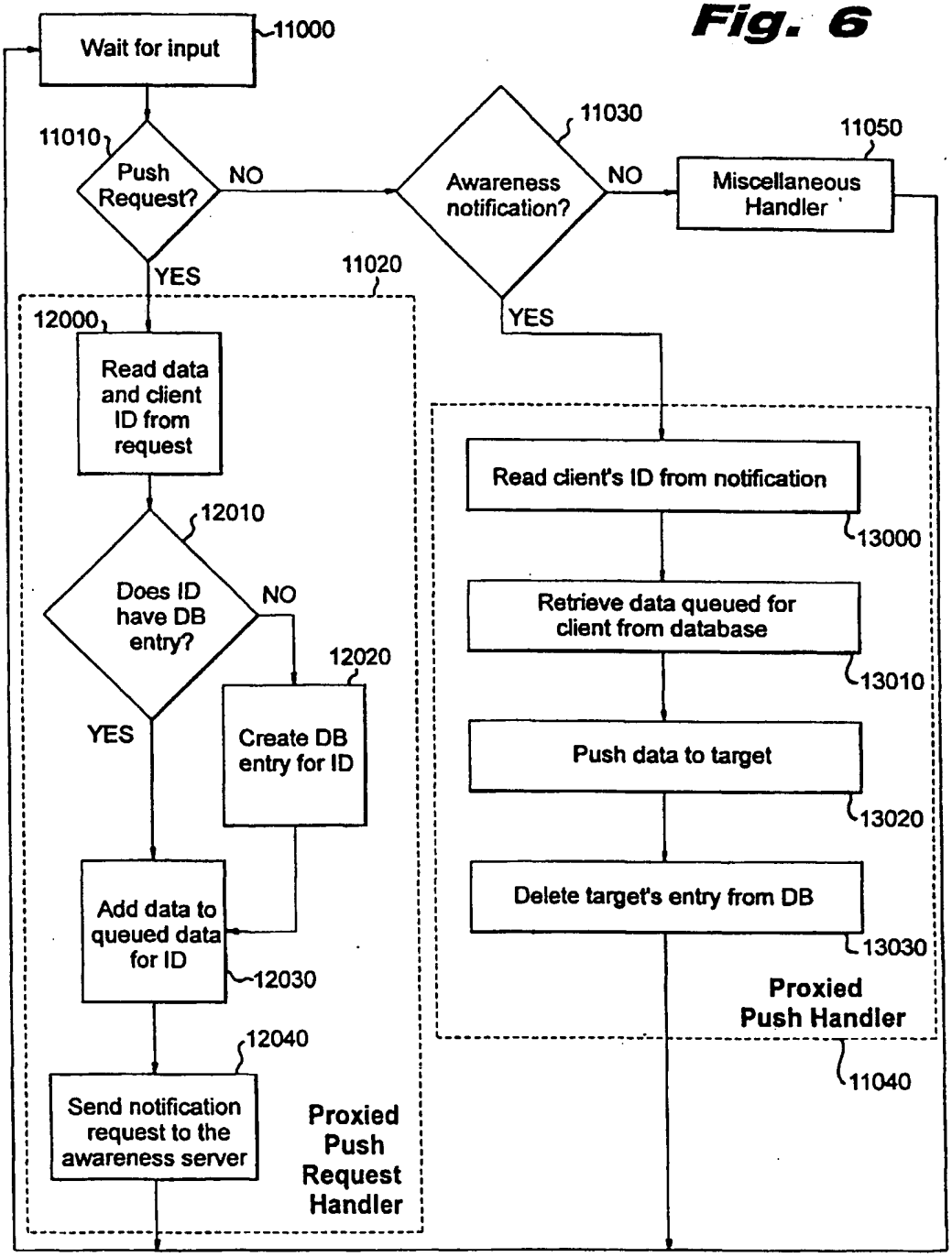

10020
Memory
10030
Proxied Push Request Handler
10040
Proxied Push Handler
10050
Proxied Push Database
CPU
10000
DASD
10010

Fig. 5

Fig. 6


Wait for input
11000
11010
Push Request?
NO
YES
11030
Awareness notification?
NO
YES
11050
Miscellaneous Handler
11020
12000
Read data and client ID from request
12010
Does ID have DB entry?
NO
YES
12020
Create DB entry for ID
Add data to queued data for ID
12030
12040
Send notification request to the awareness server
Proxied Push Request Handler
Read client's ID from notification
13000
Retrieve data queued for client from database
13010
Push data to target
13020
Delete target's entry from DB
13030
Proxied Push Handler
11040

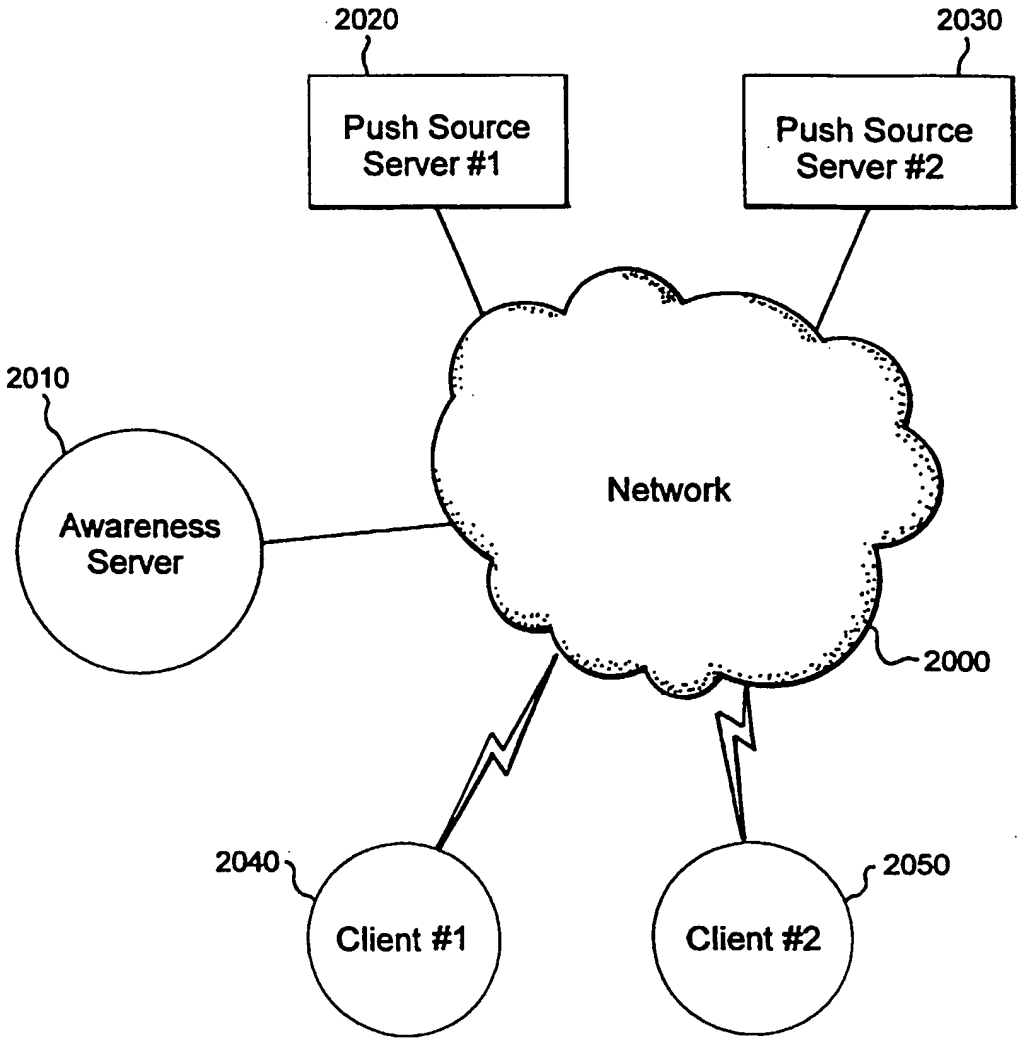

2020
2030
Push Source Server #1
Push Source Server #2
2010
Awareness Server
Network
2000
2040
2050
Client #1
Client #2

Fig. 7